Pretraživanje multimedijalnog sadržaja

Krizmanić, David

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:365199

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-27**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Stručni studij

PRETRAŽIVANJE MULTIMEDIJALNOG SADRŽAJA

Završni rad

David Krizmanić

Osijek, 2019.

Sadržaj

1. UVOD 1

1.1.Zadatak završnog rada 1

2. PRIMJENJENI PROGRAMSKI JEZIK I ALATI 2

2.1. Java programski jezik 2

2.2. Sintaksa u Javi 4

2.3. Stablo podataka 6

2.4. NetBeans 7

3. IMPLEMENTACIJA SUSTAVA 9

3.1. Dizajn korisničkog sučelja 9

3.2. Funkcionalnost i komponente aplikacije 12

4. TESTIRANJE APLIKACIJE 15

5. ZAKLJUČAK 18

LITERATURA 19

SAŽETAK 20

ABSTRACT 20

ŽIVOTOPIS 21

1. UVOD

U današnje vrijeme postoji veliki broj različitih vrsta datoteka koje ovisno o svome sadržaju imaju i određeni nastavak (ekstenzija). Nastavak datoteke označava karakteristiku sadržaja datoteke i uobičajeno je odjeljen od naziva točkom. Također nastavke datoteka koristi operacijski sustav za prepoznavanje aplikacija koje su povezane s pojedinim vrstama datoteka, npr. ako otvorimo datoteku s nastavkom "jpg" operacijski sustav traži aplikaciju koja ima mogućnost otvaranja navedene vrste datoteke.

Cilj ovog završnoga rada je upoznavanje sa Java programskim jezikom, te u konačnici kreiranje vlastite aplikacije za pretraživanje multimedijalnog sadržaja po nastavcima datoteka. U drugom poglavlju rada obrađen je teorijski dio Java programskog jezika, njegove značajke te razvoj kroz vrijeme, upoznavanje sa integriranim razvojnim okruženjem NetBeans i metoda rekurzivnog stabla. Sljedeće poglavlje detaljnije opisuje izradu korsiničkog sučelja i glavne funkcionalnosti aplikacije uz primjere koda. Zadnje, četvrto, poglavlje prikazuje način korištenja aplikacije i testira aplikaciju uz par primjera.

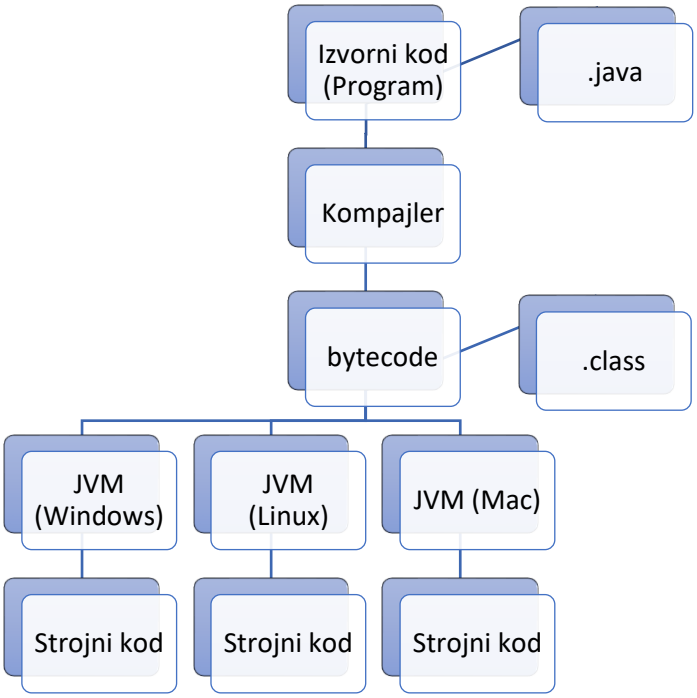
1.1. Zadatak završnog rada

Potrebno je napraviti aplikaciju za osobno računalo koja će pretraživati slike, audio i video zapise poznatijih formata na osnovu zaglavlja datoteke. Preporučena tehnologija je JAVA programski jezik.

2. PRIMJENJENE TEHNOLOGIJE I ALATI

2.1. Java programski jezik

Java [1] je programski jezik opće namjene koji je objektno orijentiran i temelji se na klasama. Projekt Java jezika pokrenuli su James Gosling, Mike Sheridan i Patrick Naughton u lipnju 1991. godine. Java je izvorno bila dizajnirana za interaktivnu televiziju, ali je tada bila previše napredna za industriju digitalne kablovske televizije. Jezik se u početku zvao Hrast po hrastu koji je stajao ispred Goslingovog ureda. Kasnije je projekt prozvan imenom Green i konačno preimenovan u Java po istoimenim zrnima kave. Gosling je Javu dizajnirao sa sintaksom u stilu C i C++ programskih jezika jer bi na taj način već bila poznata programerima sustava i aplikacija. Sun Microsystems je prvu javnu implementaciju Jave objavio pod imenom Java 1.0 1996. godine. Jedna od glavnih prednosti ovog jezika je ta što kada se jednom napiše kod on se može pokretati na svim platformama koje podržavaju Javu bez potrebe za rekompilacijom, zbog toga su koristili slogan WORA[2] (Write Once, Run Anywhere) što u prijevodu znači "Napiši jednom, pokreni bilo gdje". Java aplikacije obično se kompiliraju u bajt kod koji se može pokrenuti na bilo kojem Java virtualnom stroju (JVM) bez obzira na temeljnu računalnu arhitekturu (Slika 2.1.).



Slika 2.1. Način rada Java bytecode-a

Zadano je pet primarnih ciljeva pri stvaranju Java programskog jezika:

1. Mora biti objektno orijentirano, jednostavno i poznato.
2. Mora biti sigurno i robusno.
3. Mora biti prijenosno i neovisno o arhitekturi.
4. Mora imati mogućnost izvršavanja s visokim performansama
5. Mora biti dinamično, tumačeno i povezano.

U početku je jezik bio dosta jednostavan zbog izbacivanja mehanizama i konstrukcija korištenih od strane C-a i C++-a, a koji su mogli dovesti do grešaka. To su :

- define i typedef
- unije i strukture koje možemo realizirati klasama
- funkcije koje imaju mogućnost realiziranja statičkim metodama na razini klase
- višestruko nasljeđivanje
- naredba goto
- preopterećivanje operatora
- pokazivači, a umjesto njih koristimo reference

Javina robusnost se najviše temelji na mehanizmima rane i kasne dinamičke provjere programskog koda tijekom izvršavanja stroge sintakse i također njene provjere za vrijeme prevođenja. U javi je eliminirana bilo kakva mogućnost da dvije varijable posjeduju istu memorijsku lokaciju te tako odmah sprječava greške tijekom pisanja koda. Još jedna od ugrađenih mogućnosti kod Jave je i provjera indeksa polja i niza znakova gdje se u slučaju pogrešnog korištenja pojavljuju iznimke.. Iznimke [3] (exceptions) su Java objekti koji se kreiraju kada se u programskom kodu pojavi neka iznimna situacija kao što su npr greška korisničkog unosa, greška uređaja, fizičko ograničenje, programerska greška. Javu smatramo dinamičnim programskim jezikom jer za vrijeme rada može učitavati kod program i izvršavati ga. Također vrlo bitno za spomenuti je i činjenica da se svaki prevedeni program pohranjuje u više datoteka na disku računala tj. svaka klasa ima svoju zasebnu datoteku gdje se sprema. Ovaj način spremanja klasa omogućuje da se prevode samo dijelovi koji su izmjenjeni za razliku od C++-a gdje se sve klase prevode bez obzira na promjene.

2.2. Sintaksa u Javi

Kada govorimo o Java programu , možemo ga definirati kao skupinu objekata koji komuniciraju pozivanjem metoda. Pogledajmo sada što sve može sadržavati java sintaksa [4]:

- Klasa – Predložak kojime definiramo metode i varijable određenih objekata.
- Metoda – Metoda je zapravo ponašanje Svaka klasa može sadržavati više metoda unutar kojih se obrađuju podaci i ostale radnje.
- Varijabla instance – svaki objekt ima svoje jedinstvene varijable instance te je njegovo stanje stvoreno sa vrijednostima koje su dodane tim instancama.
- Objekt – objekti mogu imati stanja i ponašanja.
    - Stanje (auto: boja auta, ime auta, brzina auta)
    - Ponašanje (auto: ubrzavanje, usporavanje, kočenje)

Primjer jednostavnog Java programa:

```
public class MojPrviProgram {

   /* Ovo je jednostavni java program.
    * Ispisati će poruku Dobar dan!
    */

   public static void main(String []args) {
      System.out.println("Dobar dan!"); // Ispisuje Dobar dan!
   }
}
```

Programski kôd 2.1. Jednostavni Java program

Savjeti za deklaraciju varijabli/klasa/metoda:

- Osjetljivost na malo i veliko slovo – Java je osjetljiva na veliko i malo slovo, što znači da identifikatori bok i Bok imaju različita značenja.
- Imena klasa – Za sva imena klasa vrijedi da bi prvo slovo trebalo biti veliko, u slučaju da klasa sadrži više riječi treba svaku novu riječ započeti velikim slovom (npr. PrimjerKlase).
- Imena metoda – Imena metoda trebaju počinjati malim slovom dok svaku sljedeću riječ pišemo velikim (npr. public void primjerMetode()).
- Ime datoteke programa – Ime programa trebalo bi se podudarati sa imenom klase.

Svakoj Java komponenti moramo dodjeliti ime kako bi je program razlikovao od ostalih. Imena koja koristimo za varijable, metode i klase nazivamo identifikatorima. Unutar Java programa postoji nekoliko pravila za identifikatore na koje bi trebalo paziti. Neka od njih su:

- Svaki identifikator bi trebao započeti sa slovom (a-z ili A-Z), znakom valute ($) ili donjom crtom (_).
- Kao što smo već naveli i identifikatori su osjetljivi na veliko i malo slovo.
- Nakon prvog slova u imenu možemo koristiti bilo koji znak.
- Primjer dobrog imenovanja identifikatora : prezime, $novac, _ime.
- Primjer lošeg imenovanja identifikatora: 123ime, .tip, --broj.

Nakon što dodjelimo ime nekoj od varijabli isto tako moramo joj dodjeliti i tip podatka koji će predstavljati. Tipovi podataka[5] u Javi određuju razne veličine i vrijednosti koje se spremaju u varijablu. Postoje dvije vrste podataka u Javi :

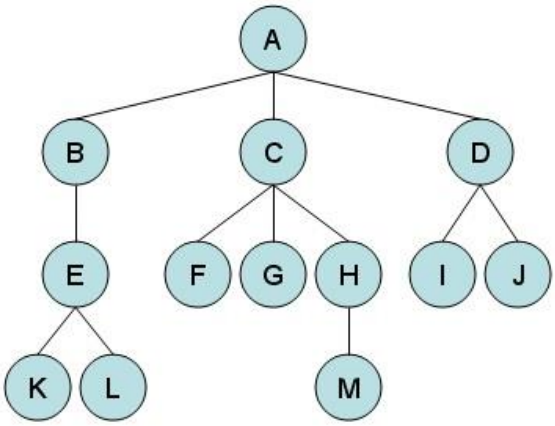
1. Primitivni tipovi podataka (boolean, char, byte, short, int, long, float i double)
2. Složeni tipovi podataka (nizovi, klase i sučelja)

Tablica 2.1. Prikaz primitivnih tipova podataka, njihovih veličina i opisa

Tip podatka	Veličina	Opis
byte	1 byte	Cijeli brojevi od -127 do 128.
short	2 bytes	Cijeli brojevi od -32,768 do 32,767.
int	4 bytes	Cijeli brojevi od -2,147,483,648 do 2,147,483,647.
long	8 bytes	Cijeli brojevi od -9,223,372,036,854,775,808 do 9,223,372,036,854,775,807.
float	4 bytes	Decimalni brojevi. Dovoljno za spremanje 6-7 decimala.
double	8 bytes	Decimalni brojevi. Dovoljno za spremanje 15 decimala
boolean	1 byte	Sprema istinite ili lažne tvrdnje.
char	2 bytes	Sprema jedan znak/slovo ili ASCII vrijednost

2.3. Stablo podataka

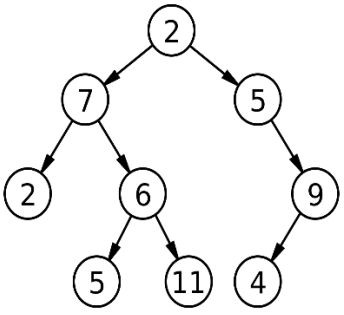
U Informatici pojam stabla predstavlja široko upotrebljavanu apstraktnu vrstu podataka koja simulira hijerarhijsku strukturu stabla[6], sa korijenskom vrijednošću i podstablima djece sa roditeljskim čvorom, predstavljenih kao skup povezanih čvorova.



Slika 2.2. Primjer strukture stable

Podatkovna struktura stable može se rekurzivno definirati kao skup čvorova (počevši od korijenskog čvora), pri čemu je svaki čvor struktura podataka koja se sastoji od vrijednosti, zajedno sa popisom referenci na čvorove uz ograničenje da niti jedna referenca nije kopija i niti jedna ne pokazuje na korijen.

Najkorištenija vrsta stabla u programiranju je binarno stablo. Binarno stablo je podatkovna struktura kojoj svaki čvora sadrži maksimalno dva djeteta (lijevo dijete i desno dijete).

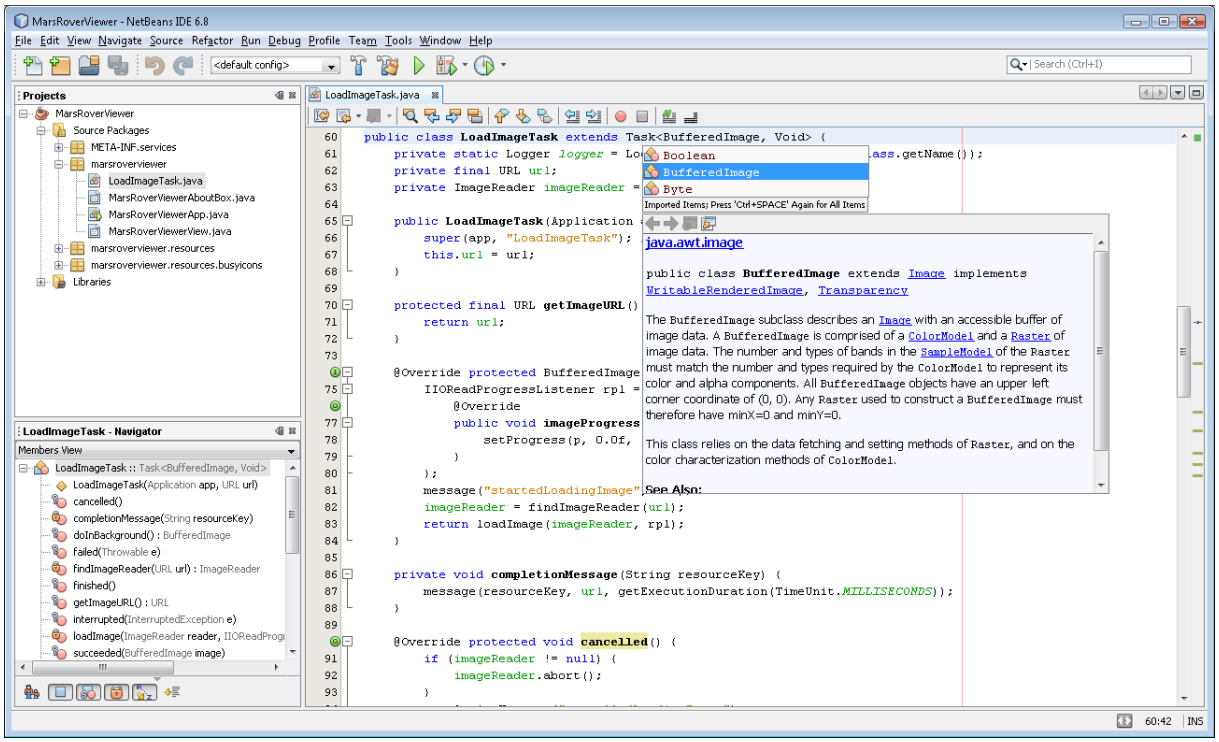


Slika 2.3. Primjer strukture binarnog stabla

2.4. NetBeans

NetBeans[7] je integrirano razvojno okruženje za Java programski jezik i prikazan je slikom 2.4. Omogućava razvoj aplikacija iz skupa modularnih softverskih komponenti koje se nazivaju moduli. NetBeans radi na više platformi (Windows, Mac, Solaris, Unix) a uz to podržava i proširenja za neke druge programske jezike kao što su: C, C++, HTML5, PHP i JavaScript.

Razvoj NetBeansa je započeo 1996. godine pod imenom Xelfi, kao studentski projekt Java integriranog razvojnog okruženja pod vodstvom Fakulteta matematike i fizike na Sveučilištu Charles u Pragu. Roman Stanek 1997. godine formira tvrtku povedenu uspjehom projekta, započinje sa komercijalnom proizvodnjom NetBeans IDE-a sve dok ga 1999. godine ne otkupljuje Sun Mycrosystems. Nakon kupnje NetBeans-a Sun odustaje od komercijalizacije te prebacuje projekt u softver otvorenog koda (open source). Sun (vlasnik NetBeans-a) 2010. godine kupljen je od strane Oracle korporacije.

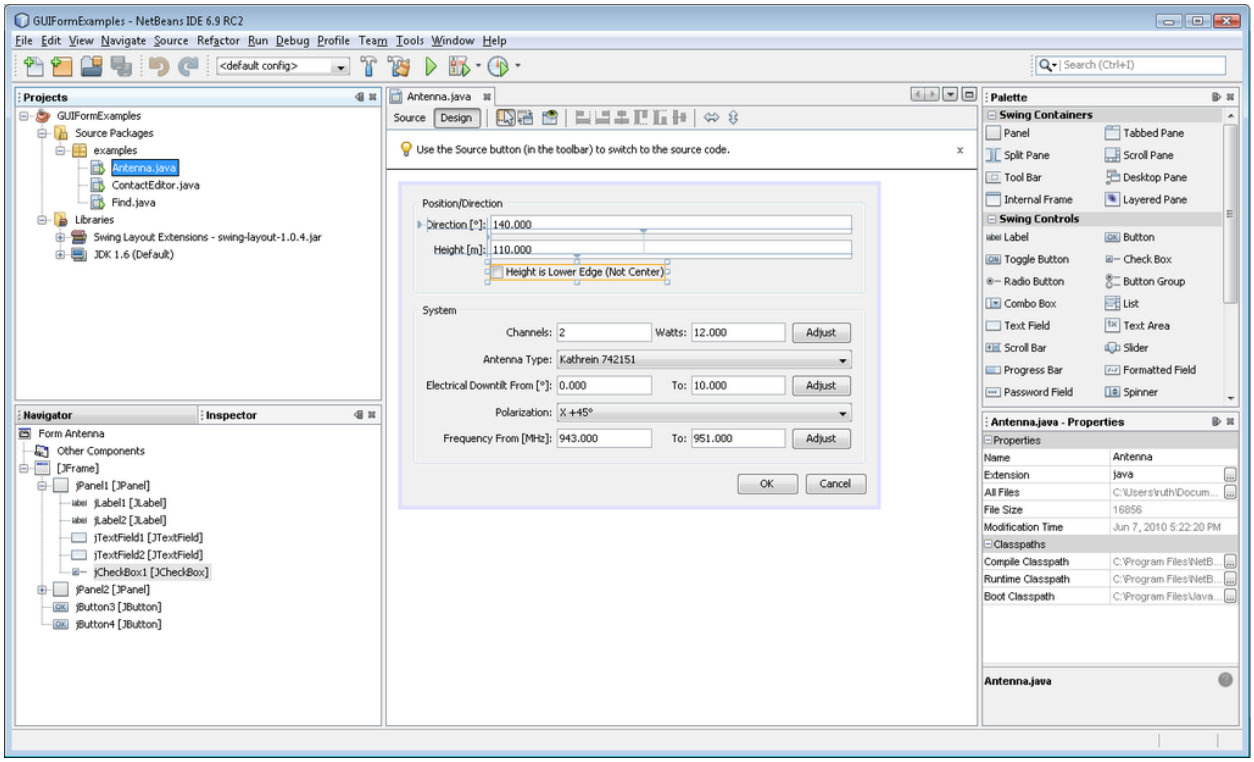


Slika 2.4. Prikaz NetBeans IDE sučelja

NetBeans platforma je okvir za pojednostavljenje razvoja Java Swing desktop aplikacija uz podršku za dinamičnu instalaciju modula . Platforma nudi usluge za višekratnu upotrebu uobičajene za desktop aplikacije, što omogućuje programerima da se usredotoče na logiku koja je specifična za njihovu primjenu.

Među glavnim značajkama platforme su :

- Upravljanje korisničkim sučeljem (izbornici i alatne trake)
- Upravljanje korisničkim postavkama
- Upravljanje skladištenjem
- Upravljanje prozorima
- Framework čarobnjak
- NetBeans vizualna knjižnica
- Integrirani razvojni alati (GUI alat prikazan na slici 2.5.)

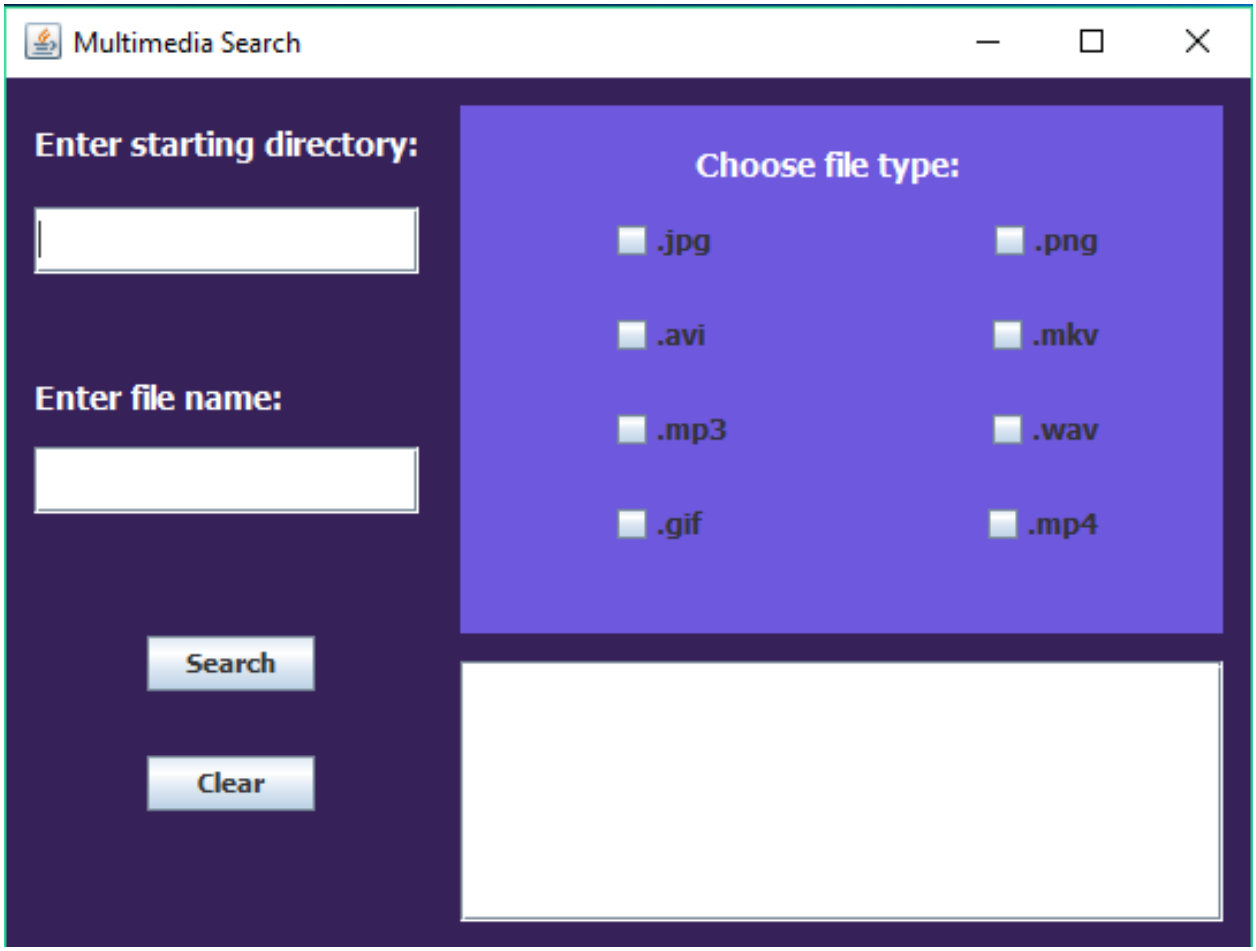


Slika 2.5. Prikaz NetBeans GUI alata za dizajn aplikacije

3. IMPLEMENTACIJA SUSTAVA

3.1. Dizajn korisničkog sučelja

Za dizajn ove aplikacije korišten je Swing GUI alat koji je integriran unutar NetBeans IDE-a te koristi povuci i spusti ("drag and drop") način pozicioniranja komponenti. Aplikacija se očituje jednostavnim dizajnom sa bojama koje odvajaju različite funkcionalne dijelove. Izgled aplikacije možemo vidjeti na slici 3.1.



Slika 3.1. Prikaz početnog zaslona aplikacije Multimedia Search

Iako skromnog dizajna, aplikacija sadržava dostatan broj Java Swing komponenata čija je inicijalizacija prikazana programskim kôdom 3.1.

```
        jPanel1 = new javax.swing.JPanel();
        jPanel3 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jTextRoot = new javax.swing.JTextField();
        jLabel2 = new javax.swing.JLabel();
        jTextName = new javax.swing.JTextField();
        jButtonSearch = new javax.swing.JButton();
        jButtonClear = new javax.swing.JButton();
        jPanel2 = new javax.swing.JPanel();
        jLabel3 = new javax.swing.JLabel();
        jCheckJpg = new javax.swing.JCheckBox();
        jCheckPng = new javax.swing.JCheckBox();
        jCheckAvi = new javax.swing.JCheckBox();
        jCheckMp3 = new javax.swing.JCheckBox();
        jCheckGif = new javax.swing.JCheckBox();
        jCheckMkv = new javax.swing.JCheckBox();
        jCheckWav = new javax.swing.JCheckBox();
        jCheckMp4 = new javax.swing.JCheckBox();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextResults = new javax.swing.JTextArea();
```

Programski kôd 3.1. Inicijalizacija Java Swing komponenata

Na sljedećim primjerima (Programski kôd 3.2., Programski kôd 3.3., Programski kôd 3.4.) biti će prikazana grafička svojstva iznad inicijaliziranih komponenata.

```
        jPanel1.setBackground(new java.awt.Color(54, 33, 89));
        jPanel1.setName("Multimedia Search");

        jPanel3.setBackground(new java.awt.Color(54, 33, 89));

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14));
        jLabel1.setForeground(new java.awt.Color(255, 255, 255));
        jLabel1.setText("Enter starting directory:");

        jLabel2.setFont(new java.awt.Font("Tahoma", 1, 14));
        jLabel2.setForeground(new java.awt.Color(255, 255, 255));
        jLabel2.setText("Enter file name:");
        jLabel2.setMaximumSize(new java.awt.Dimension(147, 17));
        jLabel2.setMinimumSize(new java.awt.Dimension(147, 17));
        jLabel2.setPreferredSize(new java.awt.Dimension(147, 17));
```

Programski kôd 3.2 Grafička svojstva Swing komponenti (Oznake i paneli)

```
        jButtonSearch.setFont(new java.awt.Font("Tahoma", 1, 11));
        jButtonSearch.setText("Search");
        jButtonSearch.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jButtonSearchMouseClicked(evt);
            }
        });

        jButtonClear.setFont(new java.awt.Font("Tahoma", 1, 11));
        jButtonClear.setText("Clear");
        jButtonClear.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jButtonClearMouseClicked(evt);
            }
        });
```

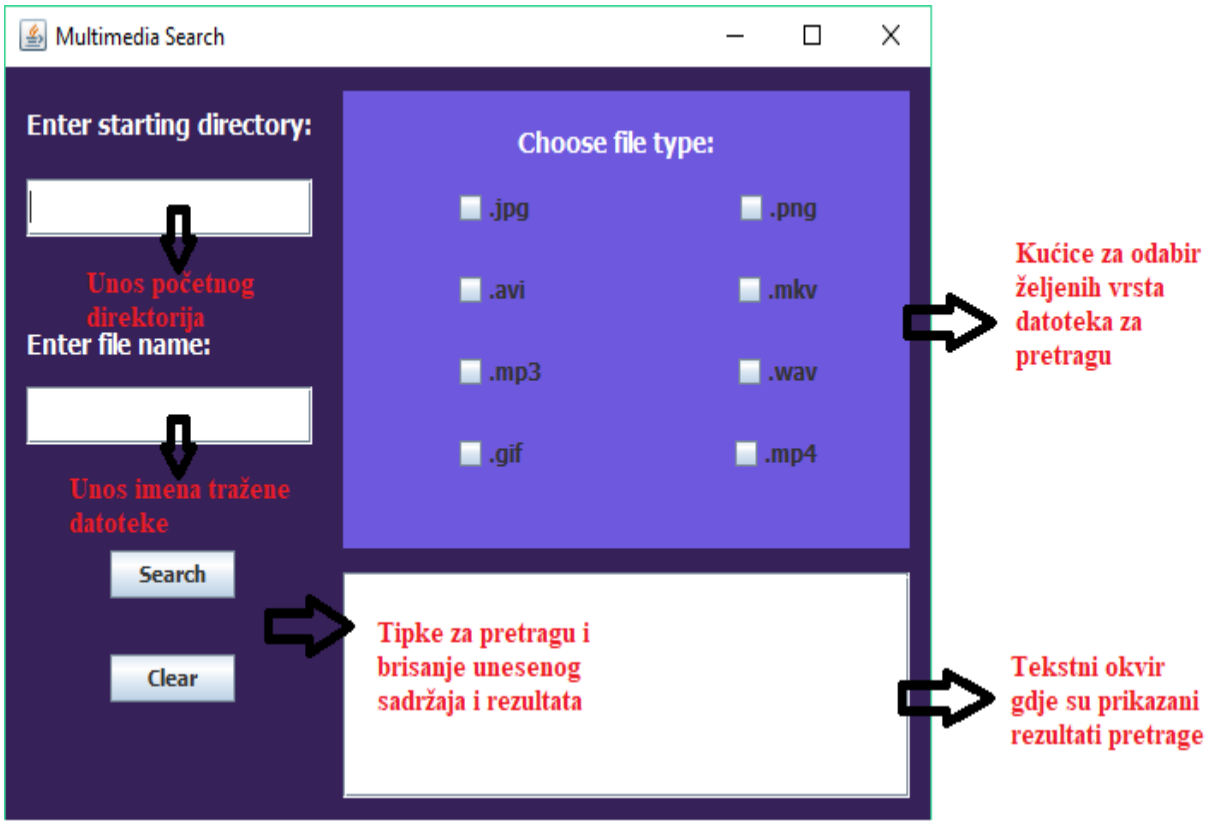
Programski kôd 3.3. Grafička svojstva Swing komponenti uz događaj pritiska miša (Tipke Search i Clear)

```
        jCheckJpg.setBackground(new java.awt.Color(110, 89, 222));
        jCheckJpg.setFont(new java.awt.Font("Tahoma", 1, 12));
        jCheckJpg.setText(".jpg");

        jCheckPng.setBackground(new java.awt.Color(110, 89, 222));
        jCheckPng.setFont(new java.awt.Font("Tahoma", 1, 12));
        jCheckPng.setText(".png");

        jCheckAvi.setBackground(new java.awt.Color(110, 89, 222));
        jCheckAvi.setFont(new java.awt.Font("Tahoma", 1, 12));
        jCheckAvi.setText(".avi");

        jCheckMp3.setBackground(new java.awt.Color(110, 89, 222));
        jCheckMp3.setFont(new java.awt.Font("Tahoma", 1, 12));
        jCheckMp3.setText(".mp3");

        jCheckGif.setBackground(new java.awt.Color(110, 89, 222));
        jCheckGif.setFont(new java.awt.Font("Tahoma", 1, 12));
        jCheckGif.setText(".gif");

        jTextResults.setColumns(20);
        jTextResults.setRows(5);
        jScrollPane1.setViewportView(jTextResults);
```

Programski kôd 3.4. Grafička svojstva Swing komponenti (Kučiće za označavanje, tekstni okvir i traka za pomicanje)

3.2. Funkcionalnost i komponente aplikacije

Aplikacija je zamišljena tako da korisnik u polje “Enter starting directory:” unosi početni direktorij od kojega želi započeti pretragu. U polje “Enter file name:” može a i ne mora unjeti traženi pojam ili dio pojma. Sljedeći korak je označavanje kućica koje sadržavaju nastavke (ekstenzije) datoteka koje korisnik ima namjeru pretražiti. Nakon unosa svih podataka i označavanja željenih kućica, klikom na tipku search izlizistavaju se rezultati pretrage u tekstni okvir. Klikom na tipku “Clear” briše se sadržaj svih polja i tekstnog okvira. Prikaz funkcionalnosti aplikacije možemo vidjeti na slici 3.2.



Slika 3.2. Prikaz funkcionalnosti aplikacije Multimedia Search

Nadalje, filitriranje i pretraživanje datoteka odrađeno je uz pomoć if iskaza kojima se provjerava koje od kućica tipova podataka su označene. Kada program provjeri koja od kućica je označena izvršava se kod unutar if izjave koji na temelju unesenog početnog direktorija uz pomoć funkcije walk prolazi kroz stablo direktorija. Dok prolazi stablom direktorija filtrira datoteke prema nastavku i unesenom imenu te ih sprema u listu stringova što je prikazano programskim kôdom 3.5.

```
 List<String> allResults = new ArrayList<String>();
         if (jCheckJpg.isSelected()){
             try (Stream<Path> walk =
Files.walk(Paths.get(jTextRoot.getText()))) {
                List<String> extensionResult = walk.map(x -
>x.toString()).filter(f -> f.endsWith(".jpg")).collect(Collectors.toList());
//Filter by extension
                 List<String> result =
getFileNamesFromListThatContains(extensionResult,jTextName.getText());
//Filter by file name
                 for(String item: result)allResults.add(item);
```

Programski kôd 3.5. Filtriranje datoteka prema nastavku i imenu datoteke

Filtriranje datoteka koje sadržavaju željeno ime ili pojam odrađen je sa metodom getFileNamesFromListThatContains. Ona uzima cijelu putanju datoteke, odvaja dijelove kod znaka "\" te uzima samo ime datoteke. Dobiveno ime datoteke uspoređuje sa onim unesenim od strane korisnika i ako ima podudranja sprema ga u listu. Primjer ove metode možemo vidjeti u programskom kôdu 3.6.

```
public List<String> getFileNamesFromListThatContains(List<String>
nameStrings,String matchingPart){
        List<String> fileNames = new ArrayList<String>();
        if(nameStrings.size()>0){
            for(String nameString :nameStrings){
                if(nameString.contains(".")){
                    String[] pathParts = nameString.split("\\\\");
                    String onlyNameString = pathParts[pathParts.length-1];
                    onlyNameString = onlyNameString.split("\\.")[0];

if(onlyNameString.contains(matchingPart))fileNames.add(nameString);
                }
            }
        }
        return fileNames;
    }
```

Programski kôd 3.6. Metoda getFileNamesFromListThatContains

Na kraju nam ostaje samo još dobivene rezultate uz pomoć for petlje spremiti u obliku string-a kako bi bili spremni za ispis u tekstnom okviru (Programski kôd 3.7.).

```
String displayResult = "";
    for(String resultString :allResults){
        displayResult += resultString + "\n";
    }
    jTextResults.setText(displayResult);
```

Programski kôd 3.7. Prikaz rezultata pretrage

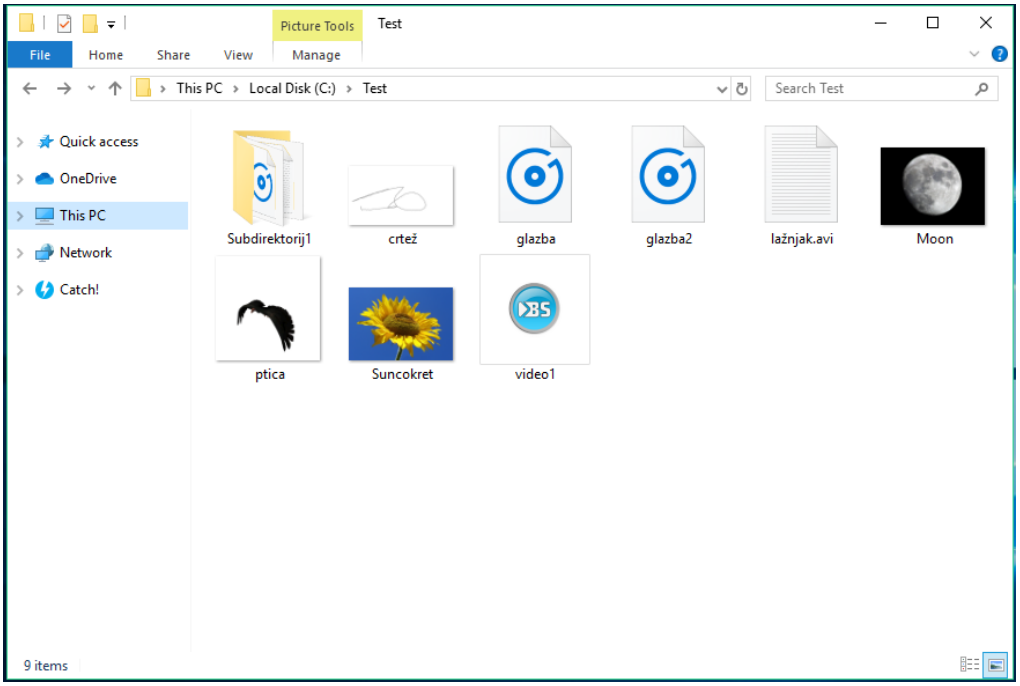
Za brisanje rezultata pretrage i unosa korisnika koristimo gumb “Clear” (Programski kôd 3.8.).

```
  private void jButtonClearMouseClicked(java.awt.event.MouseEvent evt) {
       jTextResults.setText("");
       jTextName.setText("");
       jTextRoot.setText("");
        }
```

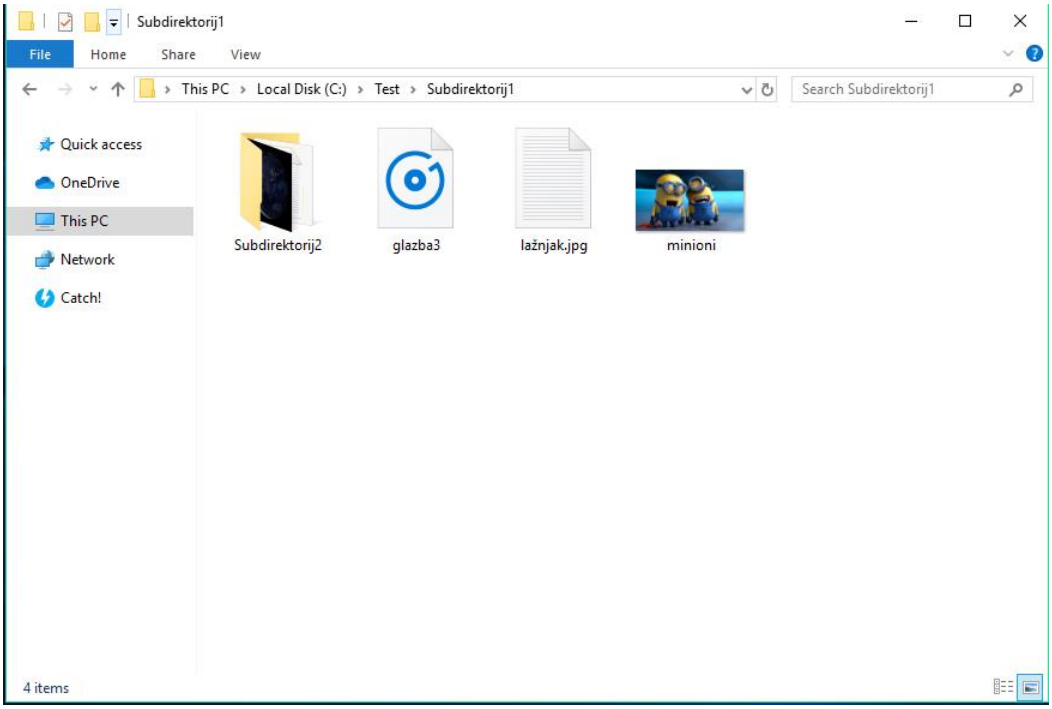
Programski kôd 3.8. Funkcija gumba Clear

4. TESTIRANJE APLIKACIJE

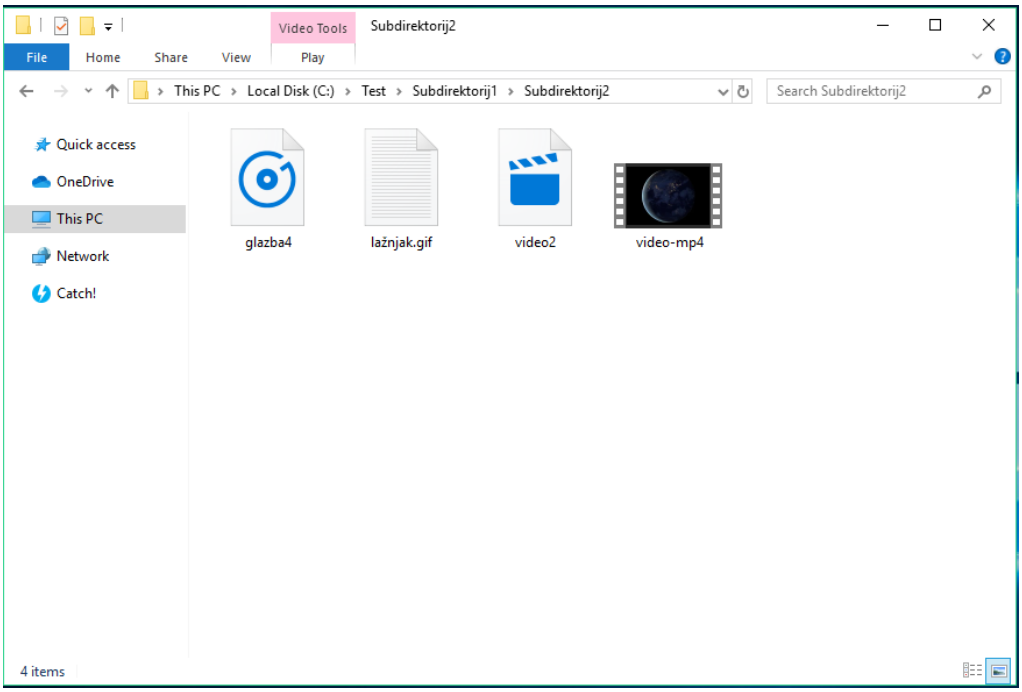
Na sljedećih nekoliko primjera biti će prikazan testni direktorij te njegovi poddirektoriji i test pretraživanja navedenog direktorija (Slika 4.1., Slika 4.2., Slika 4.3.)



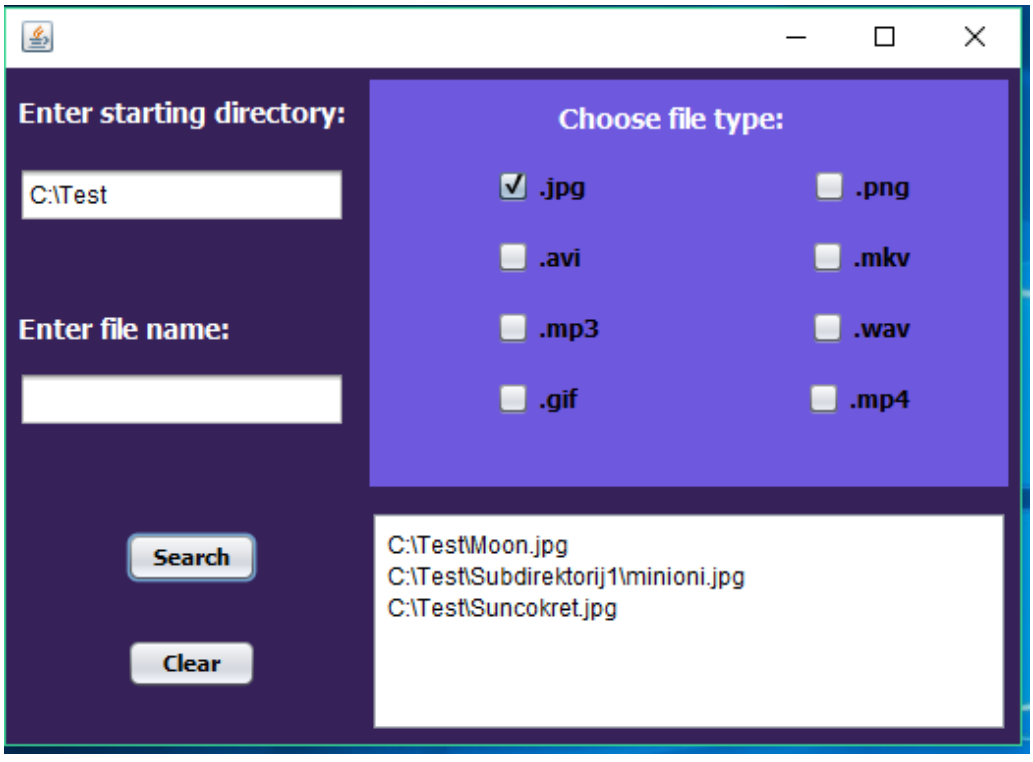
Slika 4.1. Sadržaj korijenskog direktorija "Test"



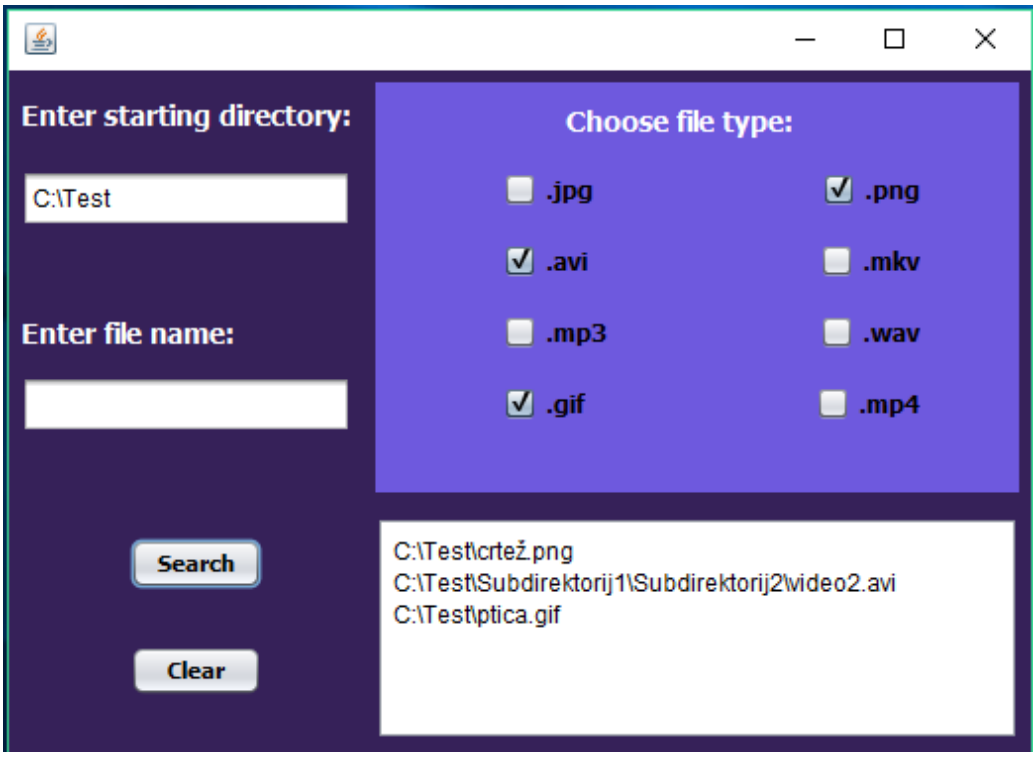
Slika 4.2. Sadržaj poddirektorija 1



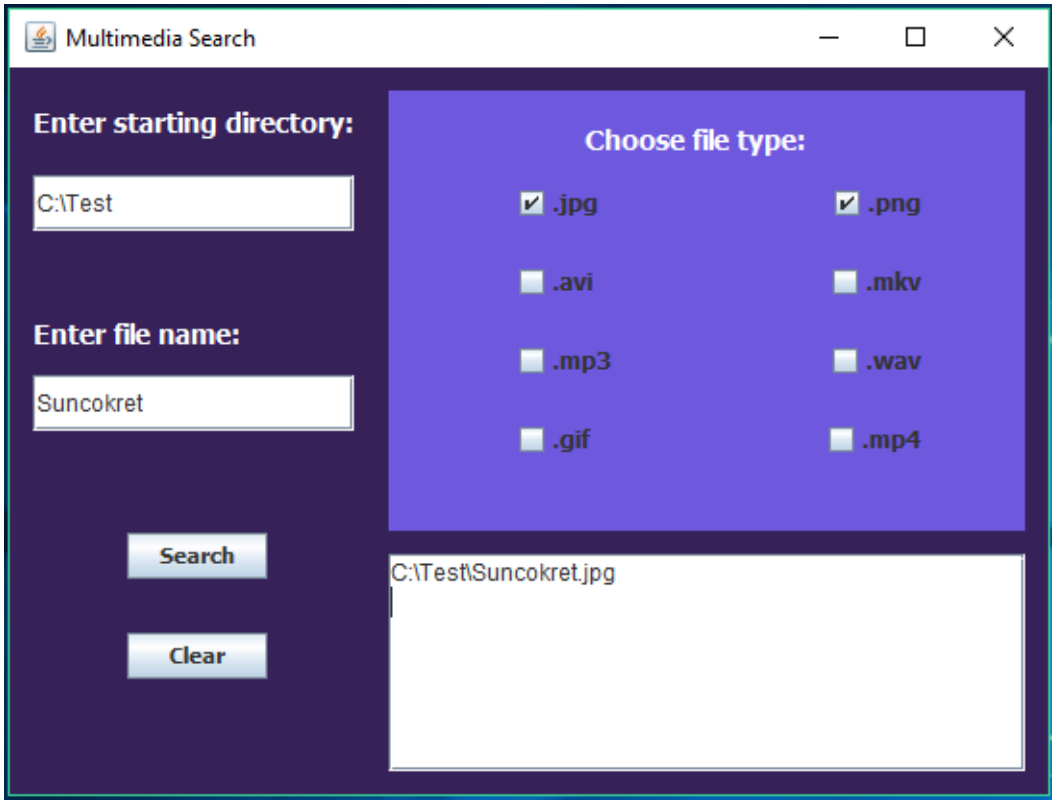
Slika 4.3. Sadržaj poddirektorija 2



Slika 4.4. Pretraga samo jedne vrste datoteka bez unosa imena



Slika 4.5. Pretraga više vrsta datoteka bez unosa imena



Slika 4.6. Pretraga više vrsta datoteka sa unosom imena

5. ZAKLJUČAK

Ovaj završni rad zahtjevao je znanje rada u Java programskom okruženju te poznavanje strukture mapa i datoteka na Windows operativnom sustavu. Sve tehnologije koje su korištene opisane su unutar završnoga rada. Prva faza izrade aplikacije bila je osmišlavanje njenog dizajna i rasporeda funkcionalnih cjelina. Druga faza je podrazumijevala pisanje koda te njegovo pridruživanje grafičkom sučelju aplikacije. Dok je treća faza služila isključivo za testiranje kreirane aplikacije.

Aplikacija za pretraživanje multimedijalnog sadržaja je jednostavna aplikacija za brzo pretraživanje mapa filitriranih po imenima i nastavcima datoteka. Omogućava korisniku unos početne mape koju želi pregledti a uz to i traženje po imenu datoteke te odabranim nastavcima. Aplikaciju odlikuje njezina jednostavnost i mogućnost traženjja više vrsta datoteka odjednom.

LITERATURA

[1] Wikipedia, Java, https://en.wikipedia.org/wiki/Java_(programming_language), rujan 2019.

[2] Wikipedia, WORA, https://en.wikipedia.org/wiki/Write_once,_run_anywhere, rujan 2019.

[3] w3schools, Java Exceptions, https://www.w3schools.com/java/java_try_catch.asp, rujan 2019.

[4] Tutorialspoint, Basic Syntax, https://www.tutorialspoint.com/java/java_basic_syntax.htm, rujan 2019.

[5] w3schools, Java Data Types, https://www.w3schools.com/java/java_data_types.asp, rujan 2019.

[6] Wikipedia, Tree (data structure), https://en.wikipedia.org/wiki/Tree_(data_structure), rujan 2019.

[7] Službena stranica NetBeansa-a, https://netbeans.org/, rujan 2019.

SAŽETAK

U završnom radu razvijena je aplikacija za pretragu multimedijalnog sadržaja uz pomoć Java programskog jezika unutar NetBeans programskog okruženja. Aplikacija omogućava korisniku unos korijenske mape od koje želi započeti pretragu, unos imena datoteke ili dijela imena te nudi odabir tipa datoteke koje će pretraživati u obliku datotečnih nastavaka. Glavni cilj ove aplikacije je brza pretraga manje razgranatog stabla mapa te prikaz dobivenih rezultata unutar aplikacije.

Ključne riječi: Java, NetBeans, pretraga, datoteka

TITLE

Multimedia content search

ABSTRACT

In this bachelor's thesis, a multimedia content search application has been developed with the help of the Java programming language using the NetBeans programming environment. The application allows the user to enter the root folder from which they want to start searching, enter a file name or part of said file name, and offers a choice of the file type to search in the form of file extensions. The main goal of this application is to quickly search less extensive folder trees and show the results obtained within the application.

Keywords: Java, NetBeans, search, file

ŽIVOTOPIS

David Krizmanić je rođen 10. listopada 1993. godine u Osijeku. Od 2000. do 2008. godine pohađa Osnonvnu školu Bilje. Godine 2008. upisuje Graditeljsko-geodetsku školu u Osijeku koju završava 2012. godine polaganjem ispita državne mature. Godine 2015. upisuje Elektrotehnički fakultet na Sveučilištu Josipa Jurja Strossmayera u Osijeku, stručni studij Elektrotehnike, smjer Informatika.

David Krizmanić